Homework 9 - Recursion

CS 1301 - Intro to Computing - Spring 2022

Important

- Due Date: **Tuesday, April 5th, 11:59 PM**.
- This is an individual assignment. High-level collaboration is encouraged, **but your submission must be uniquely yours.**
- Resources:
  - TA Helpdesk
  - Email TA's or use class Piazza
  - How to Think Like a Computer Scientist
  - CS 1301 YouTube Channel
- Comment out or delete all function calls. Only import statements, global variables, and comments are okay to be outside of your functions.
- **Read the entire document before starting this assignment.**

The goal of this homework is for you to enhance your understanding of recursion. Recall from class that recursion is when a function calls itself. Using recursion allows us to mimic using for loops (and in some cases, can even be more useful). The following homework requires the use of recursion for every problem. **NOTE: Using for/while loops instead of recursion for any of the homework problems will result in a 0 for that problem.**

Hidden Test Cases: In an effort to encourage debugging and writing robust code, we will be including hidden test cases on Gradescope for some functions. You will not be able to see the input or output to these cases. Below is an example output from a failed hidden test case:

```
Test failed: False is not true
```

Written by Paige Holland (pholland7@gatech.edu) & Audrey Cho (acho45@gatech.edu)

Page Numbers

Function Name: pageNumbers()
Parameters: bookList (`list`)
Returns: total (`int`)
Description: You just realized that you have not been reading your assigned books for English this semester! You need to finish everything before you take the final, so you want to find out the exact number of pages you have to read. Write a function that takes in a list of book page numbers (`list`) and returns the **recursive** summation of the number of pages in all of the books (`int`).

```
>>> pageNumbers([199, 434, 210])
843
```

```
>>> pageNumbers([342, 145, 302])
789
```

Letter Pyramid

Function Name: letterPyramid()
Parameters: letter (`str`), rows (`int`)
Returns: None (`NoneType`)
Description: You want to show off your programming skills to your friends before the semester is over, so you want to write a simple program that creates a pyramid. Write a function that takes in a letter (`str`) that will build the pyramid as well as the number of rows (`int`) in the pyramid. This function should **recursively** print the pyramid to the Shell. Additionally, you must ensure that a lowercase letter is passed into the function or else the function should not print anything.

```
>>> letterPyramid("a", 6)
a
aa
aaa
aaaa
aaaaa
aaaaaa
```

```
>>> letterPyramid("S", 6)
# Nothing should be printed
```

Special Character Counter

Function Name: specialChar()
Parameters: usernames (`list`)
Returns: aDict (`dict`)
Description: You want to figure out how many special characters are in some of your TA's gaming usernames. Write a function that takes in a list of usernames and returns a dictionary containing the usernames of the TAs as the keys mapped to how many special characters are in each username. For this problem, the special characters are `.`, `-`, `_`, `!`, `~`, and `#`.

Note: You can use loops to count the number of special character in the usernames (`str`), but you must use recursion to create your dictionary.

```
>>> specialChar(["cra1g_da_g0at", "arush1-f0r-da-w1n", "RAMya!", "pa1g3L#md"])
{'pa1g3L#md': 1, 'RAMya!': 1, 'arush1-f0r-da-w1n': 3, 'cra1g_da_g0at': 2}
```

```
>>> specialChar(["aryan", "fareeda4firstplace!!", "audrey.all.over.the.world"])
{'audrey.all.over.the.world': 4, 'fareeda4firstplace!!': 2, 'aryan': 0}
```

Message Decoder

Function Name: messageDecoder()
Parameters: hiddenMessage (`str`), characters (`str`)
Returns: decodedMessage (`str`)
Description: Your friend sent you a hidden message to decode! Write a function that takes in a hidden message (`str`) and a set of characters (`str`) and returns the decoded message(`str`). In order to decode a message, you need to remove all occurences of the specified set of characters, which is given as a parameter, from the hidden message string.

Note: You must use recursion to implement this function.

```
>>> messageDecoder("zdzi)N$nez)r @z) $WiLzla$Ge?z", "z$)")
"diNner @ WiLlaGe?"
```

```
>>> messageDecoder("5tcudy#incg Isn t&He# l1b#racRy&!", "c#s&")
"5tudying In tHe l1braRy!"
```

Combine Strings

Function Name: stringCombiner()
Parameters: stringList (`list`)
Returns: combinedString (`str`)
Description: Given a list containing fragmented strings, nested lists and/or tuples, write a function that **recursively** combines all string elements found within the list and returns the final combined string.

Note: Nested lists and tuples may contain strings, more nested lists and tuples, or be empty.

```
>>> stringCombiner([["Ramblin "],("Wreck"),"!"])
"Ramblin Wreck!"
```

```
>>> stringCombiner(["cs 1", ("301 i",[]), "s awes", ["ome!"]])
"cs 1301 is awesome!"
```

Grading Rubric

Function	Points
letterPyramid()	20
pageNumbers()	20
specialChar()	20
messageDecoder()	20
stringCombiner()	20
Total	**100**

Provided

The `HW09.py` skeleton file has been provided to you. This is the file you will edit and implement. All instructions for what the functions should do are in this skeleton and this document.

Submission Process

For this homework, we will be using Gradescope for submissions and automatic grading. When you submit your `HW09.py` file to the appropriate assignment on Gradescope, the autograder will run automatically. The grade you see on Gradescope will be the grade you get, unless your grading TA sees signs of you trying to defeat the system in your code. You can re-submit this assignment an unlimited number of times until the deadline; just click the “Resubmit” button at the lower right-hand corner of Gradescope. You do not need to submit your `HW09.py` on Canvas.